Modeling and Analyzing Search/Insert/Delete Problem by Petri Nets

Haiping Xu
{email: hxu1@eecs.uic.edu}
Electrical Engineering and Computer Science Department
The University of Illinois at Chicago
Chicago, IL 60607 USA

1. Problem Statement

Search/Insert/Delete Problem [1] Three kinds of processes share access to a single-linked list: searchers, inserter, and deleters. Searchers merely examine the list; hence they can execute concurrently with each other. Inserters add new items to the end of the list; insertions must be mutually exclusive to prelude two inserters from inserting new items at about the same time. However, one insert can proceed in parallel with any number of searches. Finally, deleters remove items from anywhere in the list. At most one deleter process can access the list at a time, and deletion must also be mutually exclusive with searches and insertions.

2. Modeling Search/Insert/Delete Problem by Petri Nets

Assuming we have totally k searcher, inserter and deleter, we can model the above search/insert/delete problem by the Petri net shown in Figure 1. This Petri net models a search/insert/delete synchronization, where the k tokens in place P1 represents k processes which may search, insert and delete a singly-linked list. The number of tokens in place P2, P4 and P6 represent the number of processes as searchers, deleters and inserters respectively, which are currently running. Place P3 provides mutual exclusion between searcher and deleter, while k tokens in place P3 means that at most k processes can be running as searcher/deleter at the same time. Place P5 provides mutual exclusion between deleter and inserter, while one token in place P6 means only one process can be running as deleter/inserter at any time. There are no any mutual exclusion between searchers and inserter, so searchers and inserter can be running at the same time, however only one inserter is allowed to access the singly-linked list.

A major strength of Petri nets is their support for analysis of many properties and problems associated with concurrent system. Two types of properties can be studied with a Petri net model: those which depend on the initial marking, and those which are independent of the initial marking. The former type of properties is called behavioral properties, while the latter one is called structural properties. In the following sections, we analyze both structural and behavioral properties of this Petri net, and give as many explanations as possible for the real search/insert/delete synchronization problem.

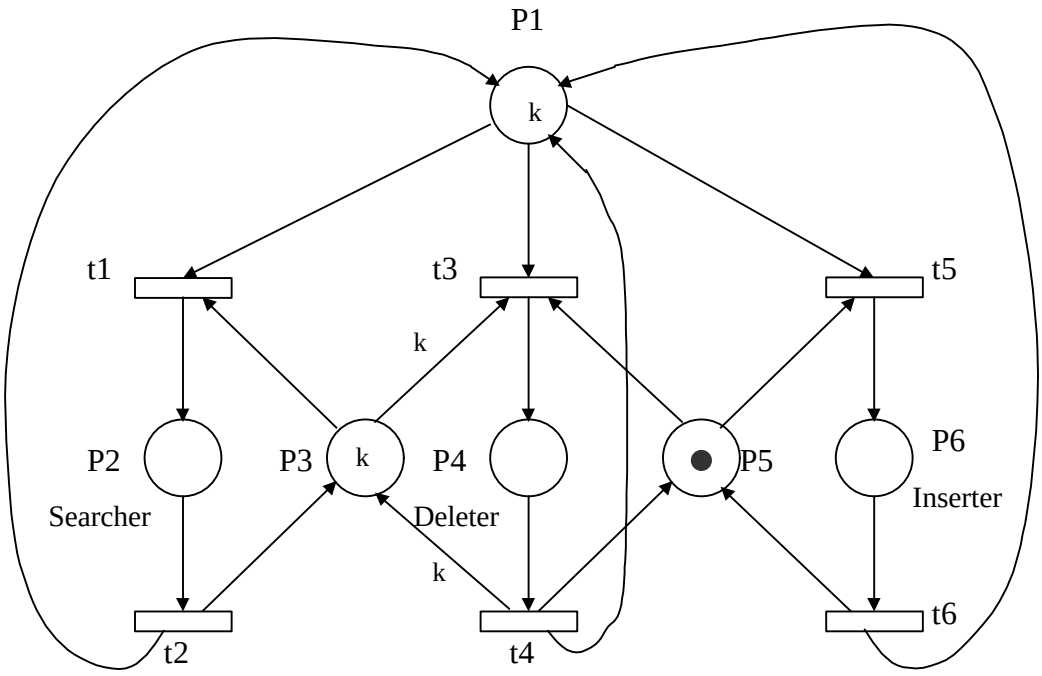

Figure 1. Petri Net for Search/Insert/Delete Problem

3. Structural Properties Analysis

Structural properties are independent of the initial marking M_0, which means these properties hold for any initial marking. To compute most of the structural properties, it's helpful to compute the minimal support S-invariants and T-invariants of the Petri net first. The incidence matrix of the Petri net in Figure 1 is given as follows:

$$A = \begin{array}{c} \\ t1 \\ t2 \\ t3 \\ t4 \\ t5 \\ t6 \end{array} \begin{array}{c} \begin{array}{cccccc} P1 & P2 & P3 & P4 & P5 & P6 \end{array} \\ \begin{bmatrix} -1 & 1 & -1 & 0 & 0 & 0 \\ 1 & -1 & 1 & 0 & 0 & 0 \\ -1 & 0 & -k & 1 & -1 & 0 \\ 1 & 0 & k & -1 & 1 & 0 \\ -1 & 0 & 0 & 0 & -1 & 1 \\ 1 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} \end{array}$$

An m-vector y of integers is called an S-invarient if Ay = 0, so the S-invariants can be computed by solving the equation Ay = 0.

We get the following minimal support S-invariants, which serve as a generator of all S-invariant:

$y1 = (1\ 1\ 0\ 1\ 0\ 1)^T$ whose minimal support || y1 || is {P1, P2, P4, P6}
$y2 = (0\ 1\ 1\ k\ 0\ 0)^T$ whose minimal support || y2|| is {P2, P3, P4}
$y3 = (0\ 0\ 0\ 1\ 1\ 1)^T$ whose minimal support || y3 || is {P4, P5, P6}

Similarly, an n-vector x of integers is called an T-invariant is $A^Tx = 0$, so the T-invariants can be computed by solving the equation $A^Tx = 0$.

We get the following minimal support T-invariants, which serve as a generator of all T-invariants:

$x1 = (1\ 1\ 0\ 0\ 0\ 0)^T$ whose minimal support || x1|| is {t1, t2}
$x2 = (0\ 0\ 1\ 1\ 0\ 0)^T$ whose minimal support || x2 || is {t3, t4}
$x3 = (0\ 0\ 0\ 0\ 1\ 1)^T$ whose minimal support || x3 || is {t5, t6}

Assume $M_0 = (k\ 0\ k\ 0\ 1\ 0)^T$, the physical interpretation of these invariants are as follows:

(1) Since $y1^TM = y1^TM_0$ for every M in $R(M_0)$, we have
M(P1) + M(P2) + M(P4) + M(P6) = k
i.e., the total number of processes in place P1, P2, P4 and P6 as an searcher/delelter/inserter is an invariant, which is always equal to k.

(2) Since $y2^TM = y2^TM_0$ for every M in $R(M_0)$, we have
M(P2) + M(P3) + k*M(P4) = k or M(P2) = M(P3) = 0 when M(P4) = 1
i.e., while an deleter is processing the list, no searchers can search the list.

(3) Since $y3^T M = y3^T M_0$ for every M in $R(M_0)$, we have
$M(P4) + M(P5) + M(P6) = 1$
i.e., either none of deleter/inserter is processing the list or only one of them is in place P4/P6.

(4) Since $M = M_0 + A^T x1 = M_0$, the T-invariant $x1=(1\ 1\ 0\ 0\ 0\ 0)^T$ says that the initial marking M_0 is reachable from the same M_0 after firing t1 and t2 once. This is true in Figure 1, and it means a searcher searched the list and returned back to place P1 after done.

(5) Since $M = M_0 + A^T x2 = M_0$, the T-invariant $x1=(0\ 0\ 1\ 1\ 0\ 0)^T$ says that the initial marking M_0 is reachable from the same M_0 after firing t3 and t4 once. This is true in Figure 1, and it means that a deleter deleted an element from the list and returned back to place P1 after done.

(6) Since $M = M_0 + A^T x1 = M_0$, the T-invariant $x1=(0\ 0\ 0\ 0\ 1\ 1)^T$ says that the initial marking M_0 is reachable from the same M_0 after firing t5 and t6 once. This is true in Figure 1, and it means that an inserter inserted an element to the list and returned back to place P1 after done.

The following equation gives an upper bound on the number of tokens that place P can ever have, here the minimum is taken over all nonnegative minimal-support S-invariants y_i:

$$M(P) \le \text{Min} [\ M_0^T y_i / y_i(P)\ ]$$

For instance,

$M(P2) \le \text{Min}[\ M_0^T y1 / y1(P2),\ M_0^T y2 / y2(P2),\ M_0^T y3 / y3(P2)\ ]$
$= \text{Min}[\ k / 1,\ k / 1,\ 1 / 0\ ]$
$= k$

which means k searchers can search the list concurrently;

$M(P4) \le \text{Min}[\ M_0^T y1 / y1(P4),\ M_0^T y2 / y2(P4),\ M_0^T y3 / y3(P4)\ ]$
$= \text{Min}[\ k / 1,\ k / k,\ 1 / 1\ ]$
$= 1$

which means only one deleter can process the list at any time;

$M(P6) \le \text{Min}[\ M_0^T y1 / y1(P6),\ M_0^T y2 / y2(P6),\ M_0^T y3 / y3(P6)\ ]$
$= \text{Min}[\ k / 1,\ k / 0,\ 1 / 1\ ]$
$= 1$

which means only one inserter can process the list at any time.

All these boundedness properties are consistent with the nature of our Search/Insert/Delete problem. Now, lets summarize ten structural properties for the Petri net shown in Figure 1.

Structural Boundedness

Since we have $y = y1 + y2 + y3 = (1\ 2\ 1\ k+2\ 1\ 1\ 2)^T > 0$ and $Ay = 0 \leq 0$, this Petri net is structurally bounded. It means that each place in this Petri net is bounded for any finite initial marking M_0.

Conservativeness

Since we have $y = (1\ 2\ 1\ k+2\ 1\ 1\ 2)^T > 0$, and $Ay = 0$, this Petri net is conservative. It means that there exists a positive integer y(P) for every place P such that the weighted sum of tokens, $M^Ty = M_0{}^Ty$ is a constant for every $M \in R(M_0)$ and for any fixed initial marking M_0.

Partial Conservativeness

Since we have $y = (1\ 2\ 1\ k+2\ 1\ 1\ 2)^T \geq\neq 0$, and $Ay = 0$, this Petri net is partially conservative. It means that there exists a positive integer y(P) for some place P such that the weighted sum of tokens, $M^Ty = M_0{}^Ty$ is a constant for every $M \in R(M_0)$ and for any fixed initial marking M_0. Actually, this net must be partially conservative because it is conservative.

Repetitiveness

Since we have $x = x1 + x2 + x3 = (1\ 1\ 1\ 1\ 1\ 1)^T > 0$, and $A^Tx = 0 >= 0$, this Petri net is repetitive. It means that there exists an initial marking M_0 and a firing sequence σ from M_0 such that every transition occurs infinitely often in σ. In our example, we have the firing sequence $\sigma = <<t1, t2, t3, t4, t5, t6>^{\infty}>$ in which every transition occurs infinitely often.

Partial Repetitiveness

Since we have $x = x1 + x2 + x3 = (1\ 1\ 1\ 1\ 1\ 1)^T \geq\neq 0$, and $A^Tx = 0 >= 0$, this Petri net is partially repetitive. It means that there exists an initial marking M_0 and a firing sequence σ from M_0 such that some transition occurs infinitely often in σ. Actually, this Petri net must be partially repetitive because it is repetitive.

Consistency

Since we have $x = x1 + x2 + x3 = (1\ 1\ 1\ 1\ 1\ 1)^T > 0$, and $A^Tx = 0$, this Petri net is consistent. It means that there exists an initial marking M_0 and a firing sequence σ from M_0 back to M_0 such that every transition occurs at least once in σ. In our example, we have the firing sequence $\sigma = <t1, t2, t3, t4, t5, t6>$. By firing σ, the initial marking M_0 can be reproduced and every transition occurs at least once in σ.

Partial Consistency

Since we have $x = x1 + x2 + x3 = (1\ 1\ 1\ 1\ 1\ 1)^T \geq 0$, and $A^T x = 0$, this Petri net is partially consistent. It means that there exists an initial marking M_0 and a firing sequence σ from M_0 back to M_0 such that some transition occurs at least once in σ. Actually, this net must be partially consistent because it is consistent.

Controllability

Since marking $M = (0\ 1\ 1\ 0\ 0\ 0)^T$ is not reachable from the initial marking $M_0 = (k\ 0\ k\ 0\ 1\ 0)^T$, this Petri net is not completely controllable.

Structural Liveness

Since there exists a live initial marking $M_0 = (k\ 0\ k\ 0\ 1\ 0)^T$ for this Petri net, it is structurally live.
Actually, this property can be drawn from the following theorem (remains to be proved):

Theorem: A Petri net is structurally live if every siphon has a trap.

In Figure 1, we have the following set of siphons and traps:

Siphons:

S1 = {P2, P3, P4} minimal, basis

S2 = {P4, P5, P6} minimal, basis

S3 = {P1, P2, P4, P6} minimal, basis

S4 = {P2, P3, P4, P5, P6} = S1 ∪ S2

S5 = {P1, P2, P3, P4, P5, P6} = S1 ∪ S2 ∪ S3

Traps:

Q1 = {P2, P3, P4} minimal, basis

Q2 = {P4, P5, P6} minimal, basis

Q3 = {P1, P2, P4, P6} minimal, basis

Q4 = {P2, P3, P4, P5, P6} = Q1 ∪ Q2

Q5 = {P1, P2, P3, P4, P5, P6} = Q1 ∪ Q2 ∪ Q3

Since each siphon in the Petri net of Figure 1 has a trap, from the above theorem, we know that this Petri net is structurally live.

Structural B-Fairness

Since there are more than one reproduction vectors and this Petri net is structurally bounded and consistent, we know that it is not structural B-fair. Actually, for initial marking $M_0 = (k\ 0\ k\ 0\ 1\ 0)^T$, we have an infinite firing sequence $\{ <t1, t2>^{\infty} \}$, and transitions t3 and t5 are never fired.

We have computed ten structural properties for the Petri net in Figure 1. In the next section, we will talk about the behavior properties for this Petri net.

4. Behavioral Properties Analysis

Behavioral properties are marking dependent properties. In this section, we will analyze seven types of behavioral properties for the Petri net in Figure 1. These properties are: reachability, boundedness, liveness, reversibility, coverability, persistence and fairness.

Reachability

A marking M_n is said to be reachable from a marking M_0 if there exists a sequence of firings that transforms M_0 to M_n. The reachability of a Petri net can be analyzed by drawing its reachability graph. To make the graph looks simple and sufficient for our behavioral properties analysis, we assume k = 2. The reachability graph for the Petri net in Figure 1 is shown in Figure 2.

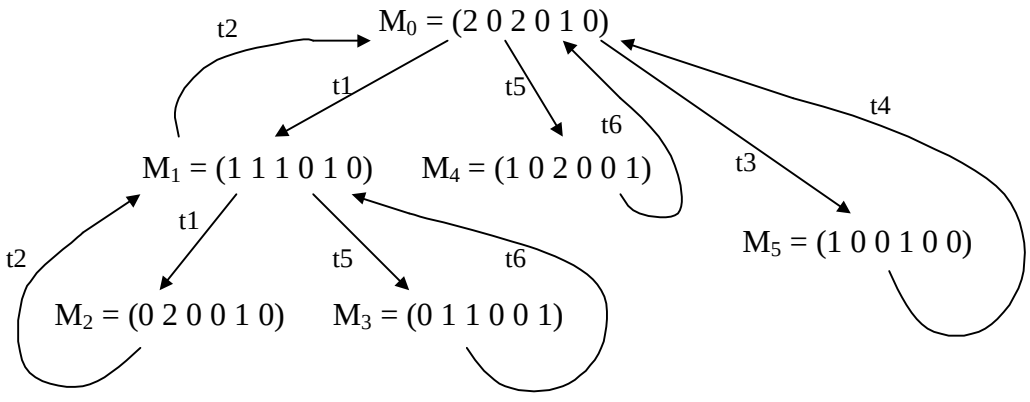

Figure 2. Reachability Graph for the Perti Net in Figure 1

From the above reachability graph, we can see that any marking $M_n \in R(M_0)$ is reachable from any marking $M_n' \in R(M_0)$. For instance, marking M5 = (1 0 0 1 0 0) can be reached from M3 = (0 1 1 0 0 1) by firing t6, t2 and t3 once. Also, we observed the following properties:

$M(P2) + M(P4) \leq 1$ for any marking $M \in R(M_0)$
$M(P4) + M(P6) \leq 1$ for any marking $M \in R(M_0)$

This is because of the mutual exclusion between searcher and deleter, and similarly between deleter and inserter as well. However, we have M3(P2) + M3(P6) = 2. This property means that searcher and inserter can process the list concurrently. Finally, we have $M(P4) \leq 1$ and $M(P6) \leq 1$ for any $M \in R(M_0)$, however we have M2(P2) = 2, which indicates that two searchers can process the list at the same time, while only one deleter/inserter is allowed to process the list at any time.

Boundedness

The boundedness of this Petri net is obvious by investigating the reachability graph in Figure 2. In the case of k = 2, we find that $M(P) \leq 2$ for every place P in the net and for every marking $M \in R(M_0)$, so this Petri net is 2-bounded. In the case of k being any fixed positive number, it's easy for us to conclude that this Petri net is k-bounded.

Liveness

Since the Petri net N in Figure 1 is structural live, we must have an initial marking M_0 for N, such that (N, M_0) is live. This behavioral property guarantee deadlock-free operations. No matter what firing sequence is chosen, the searcher/deleter/inserter problem will never be deadlocked.

Reversibility

A Petri net (N, M_0) is said to be reversible if, for each marking M in $R(M_0)$, M_0 is reachable from M. From the reachability graph in Figure 2, it's easy to see that M_0 = (2 0 2 0 1 0) is reachable from any marking $M \in R(M_0)$, so this Petri net is reversible. Similarly, this Petri net is also reversible in the case of M_0 = (k 0 k 0 1 0), where k > 2.

Coverability

A marking M in a Petri net (N, M_0) is said to be coverable if there exists a marking M1 in $R(M_0)$ such that $M1(P) \geq M(P)$ for each P in the net. Let M_{min} be the minimum marking needed to enable a transition t in (N, M_0), it's easy to see that t is L1-live in (N, M0) if and only if M_{min} is coverable.

In our example, we still assume k = 2, since for transition t1, M_{min} = (1 0 1 0 0 0), and it can be covered by M1 = (1 1 0 1 0), so transition t1 is L1-live. Similarly, transition t3 is L1-live because its M_{min} = (1 0 2 0 1 0) can be covered by M_0 = (2 0 2 0 1 0). All other transitions in this Petri net can be proved to be L1-live in the same way.

Persistence

This Petri net (N, M_0), where M_0 = (k 0 k 0 1 0) , is not persistent because two enabled transitions t3 and t5, the firing of one transition will disable the other one.

This is because transitions t3 and t5 have a common input place P1, they are called in a structural conflict; moreover, this pair of transitions is also in a behavioral conflict because for a reachable marking M = (k 0 k 0 1 0), both are enabled, but firing one disables the other. We have the conclusion that a Petri net is nonpersistent if and only if it has a pair of transitions in a behavioral conflict.

Fairness

This Petri net is not bounded fair (BF), unconditionally fair (UF) and strongly fair (SF) because we have an infinite firing sequence $\sigma = < <t1, t2>^{\infty} >$, where t3 and t5 are enabled infinitely often, however they are never fired.

When k = 1, this Petri net is weakly fair (WF) because no transitions in the net is enabled continuously. However, when k > 1, for the infinite firing sequence $\sigma = < <t1, t2>^{\infty} >$, t3 and t5 are enabled continuously, but they are never fired, so σ is not weakly fair, which implies that the Petri net (N, M_0), where M_0 = (k 0 k 0 1) and k > 1, is not weakly fair.

In our real problem, the unfairness of our Petri net means that searcher/deleter/inserter may starve to death.

We have talked about the behavioral properties of this Petri net. In the last section, we will investigate the preservation of structural liveness and boundedness through net reduction.

5. Preservation of Structural LB through Net Reduction

The two linear dependence rules for net reduction are listed as follows[2]:

Rule1: The reduction rule ϕP: This transformation is the removal of a nonnegatively linearly dependent place p together with it's incident arcs.

Rule2: The reduction rule ϕT: This transformation is the removal of a nonnegatively linearly dependent transition t together with it's incident arcs.

In addition to these two rules, for our purpose, we also need to use the six simple reduction rules which preserve liveness, safeness and boundedness for any unrestricted Petri net. These simple reduction rules are described in Prof. Tadao Mutata's lecture notes[2], which are Fusion of Series Places (FSP), Fusion of Series Transitions (FST), Fusion of Parallel Places (FPP), Fusion of Parallel Transitions (FPT), Elimination of Self-loop Places (ESP), and Elimination of Self_lop Transitions (EST).

The following theorem is now advanced but it still remains to be proved:

Theorem: Any structurally live and bounded Petri net N with a live marking M_0 can be reduced to an structurally live and bounded Petri net N' with a living marking M_0' by the application of some rules of the set {ϕP, ϕT, FSP, FST, FPP, FPT, ESP, EST}.

By examining our example, we may verify the above theorem by net reduction with the structural liveness and boundedness properities preserved.

The incidence matrix of the Petri net in Figure 1 is rewritten as follows:

$$A = \begin{array}{c} \\ t1 \\ t2 \\ t3 \\ t4 \\ t5 \\ t6 \end{array} \begin{array}{c} \begin{array}{cccccc} P1 & P2 & P3 & P4 & P5 & P6 \end{array} \\ \begin{bmatrix} -1 & 1 & -1 & 0 & 0 & 0 \\ 1 & -1 & 1 & 0 & 0 & 0 \\ -1 & 0 & -k & 1 & -1 & 0 \\ 1 & 0 & k & -1 & 1 & 0 \\ -1 & 0 & 0 & 0 & -1 & 1 \\ 1 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} \end{array}$$

Since c(P1) = c(P2) + 2*c(P3) + 2k*c(P4) + c(P5), by using reduction rule ϕP, we can remove place P1, and the resulting Petri net is shown in Figure 3.

Now we can use FST to merge t1 and t2, t5 and t6, and use EST to eliminate two self-loop transitions, which are t12 and t56. The result of our reduced Petri net is shown in Figure 4. Finally, by firing t3, and then use FPP k times, we get our final version of our reduced net model shown in Figure 5. Obviously, the Petri net in Figure 5 is structural live and bounded, and the theorem advanced above was verified as so.

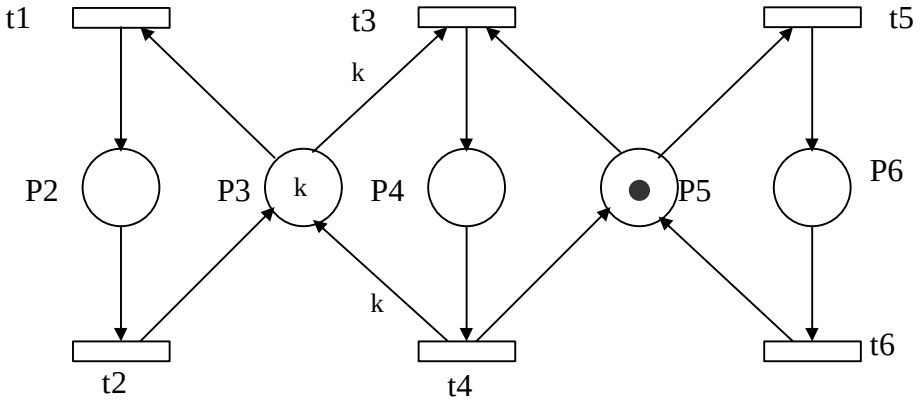

Figure 3. Petri Net for Search/Insert/Delete Problem reduced by rule ϕP

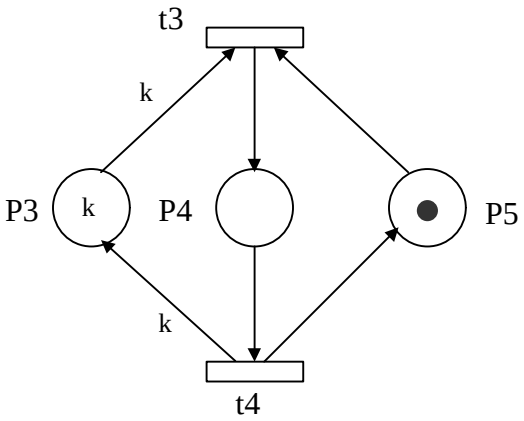

Figure 4. Petri Net for Search/Insert/Delete Problem reduced by ϕP, FST and EST

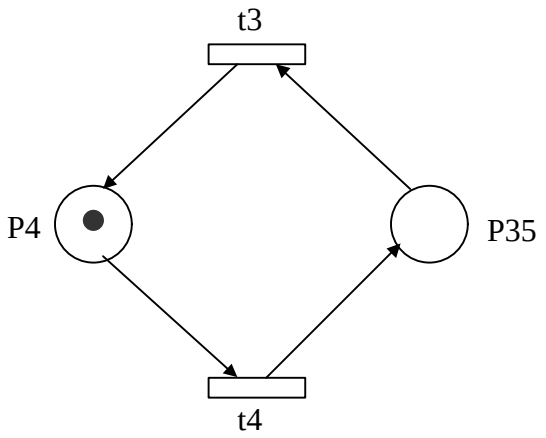

Figure 5. Petri Net for Search/Insert/Delete Problem reduced by ϕP, FST, EST and FPP

6. Conclusion

Petri nets are powerful for modeling concurrent, non-determinism, synchronization and mutual exclusion. In this paper, we first modeled the searcher/deleter/inserter problem by Petri net, then we summarized the structural properties and behavioral properties for this Petri net model. Finally, a theorem for net reduction was advanced and verified by our chosen problem.

Reference:

[1] G.R. Andrews, *Concurrent Programming: principles and practice*, Addison-Wesley Publishing Company, 1991.

[2] Tadao Murata, *Petri Net Modeling and Analysis of Concurrent Systems*, EECS 564 Lecture Notes, University of Illinois at Chicago, 1999.